Beginning Flash® Game Programming FOR DUMMIES®

by Andy Harris

Wiley Publishing, Inc.

Beginning Flash® Game Programming FOR DUMMIES®

by Andy Harris

Wiley Publishing, Inc.

Beginning Flash® Game Programming For Dummies®
Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774

www.wiley.com

Copyright © 2006 by Wiley Publishing, Inc., Indianapolis, Indiana

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, or online at http://www.wiley.com/go/permissions.

Trademarks: Wiley, the Wiley Publishing logo, For Dummies, the Dummies Man logo, A Reference for the Rest of Us!, The Dummies Way, Dummies Daily, The Fun and Easy Way, Dummies.com, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 800-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

For technical support, please visit www.wiley.com/techsupport.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Control Number: 2005927728

ISBN-13: 978-0-7645-8962-1

ISBN-10: 0-7645-8962-8

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

1O/TR/RQ/QV/IN

About the Author

Andy Harris earned a degree in Special Education from Indiana University/Purdue University–Indianapolis (IUPUI). He taught young adults with severe disabilities for several years. He also taught himself enough computer programming to support his teaching habit with freelance programming. Those were the exciting days when computers started to have hard drives, and some computers connected to each other with arcane protocols. He taught programming in those days because it was fun.

Eventually, Andy decided to teach computer science full time, and he still teaches at IUPUI. He lectures in the applied computing program and runs the streaming media lab. He also teaches classes in whatever programming language is in demand at the time. He has developed a large number of online video-based courses and international distance education projects.

Andy has written several books on various computing topics and languages including Java, C#, mobile computing, JavaScript, and PHP/MySQL.

Andy welcomes comments and suggestions about his books. He can be reached at `aharris@cs.iupui.edu`.

Dedication

This book is dedicated to Heather, Elizabeth, Matthew, Jacob, and now Benjamin.

Author's Acknowledgments

Although writing a book often seems like a lonely endeavor, it really takes a lot of talented and dedicated people to make a book on a topic as complex as this. Fortunately, I am blessed by my companions in this process.

First, I give thanks to Him from whom all flows.

Even nonfiction books have heroes. My hero is my wife, Heather. You are the unending delight of my life. Thank you for who you are and for all you do. Thanks also to all my kids. I know it's rough when Daddy spends so much time writing. I'm done for a little while. Let's go play! I love you guys.

Thanks to my dear friend Melody Layne who helped me once again take my writing career to a new place.

Thanks to acquisitions editor Katie Feltman. Even when I couldn't find the restaurant where we had our first meeting, you had faith in me and encouraged me all through the process. Thanks also to project editor Pat O'Brien, the Drill Sergeant For Dummies who can turn even me into an author worthy of the incredible *For Dummies* series. I'm still learning, Pat. One day, I'll really get it. Seriously, thanks for all the guidance. I really appreciate learning from you and working with you. Another big thank you goes to copy editor Teresa Artman: I'm amazed how she can take my mush and turn it into something that actually sounds good. And a big thanks to Scott Hofmann for technical editing.

The production process that goes behind a book is dizzying and impressive. I'd like to thank everyone at Wiley for their professionalism. The folks in layout, composition, graphics, proofing, cover work, marketing, and everyone else who worked on this book all deserve three cheers for their terrific work.

Thanks to Macromedia for developing Flash in a way that is adaptable for games and accessible to people who are not wealthy.

Thanks to John Gersting for looking over my code and giving me such good advice and guidance.

A very special thanks to my students, especially those in Web Game Development N451. You always teach me way more than I can ever teach you. Thank you for letting me be your teacher.

Publisher's Acknowledgments

We're proud of this book; please send us your comments through our online registration form located at www.dummies.com/register/.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial, and Media Development

Senior Project Editor: Pat O'Brien

Acquisitions Editor: Katie Feltman

Senior Copy Editor: Teresa Artman

Technical Editor: Scott Hofmann

Editorial Manager: Kevin Kirschner

Media Development Specialist: Laura Moss

Media Development Manager: Laura VanWinkle

Media Development Supervisor: Richard Graves

Editorial Assistant: Amanda Foxworth

Cartoons: Rich Tennant (www.the5thwave.com)

Composition Services

Project Coordinator: Adrienne Martinez

Layout and Graphics: Carl Byers, Andrea Dahl, Joyce Haughey, Stephanie D. Jumper, Barbara Moore, Barry Offringa, Lynsey Osborn

Proofreaders: Leeann Harney, Jessica Kramer, Joe Niesen, Carl William Pierce, Rob Springer, TECHBOOKS Production Services

Indexer: TECHBOOKS Production Services

Special Help: Rebecca Senninger

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Diane Graves Steele, Vice President and Publisher

Joyce Pepple, Acquisitions Director

Composition Services

Gerry Fahey, Vice President of Production Services

Debbie Stailey, Director of Composition Services

Contents at a Glance

Introduction 1

Part I: Basic Flash 7

Chapter 1: Why You Want to Write Games in Flash 9
Chapter 2: Cruising and Using the Flash Environment 19

Part II: The Next Steps 43

Chapter 3: Altered States 45
Chapter 4: Getting with the Program 61
Chapter 5: Making an Interactive Game 85

Part III: Sprites, or Movie Clips 111

Chapter 6: Introducing Sprites and Movie Clips 113
Chapter 7: Won't Be Long 'Til You Write Pong 141

Part IV: Getting Control of the Situation 173

Chapter 8: Keyboard Input and Audio Output 175
Chapter 9: It's Alive! Animating Your Sprites 193
Chapter 10: Building the Monster Traffic Game 219

Part V: Phun with Phuzzy Physics 245

Chapter 11: Vectors and Gravity 247
Chapter 12: Vehicle Motion 279
Chapter 13: The Life and Death of Sprites 317

Part VI: The Part of Tens 341

Chapter 14: Ten Math Concepts for Game Programmers 343
Chapter 15: Ten Game Starters 351

Index 377

Table of Contents

Introduction 1

What's Really (Not) Required 1
About This Book 2
How This Book Is Organized 3
Part I: Basic Flash 3
Part II: The Next Steps 3
Part III: Sprites, or Movie Clips 3
Part IV: Getting Control of the Situation 3
Part V: Phun with Phuzzy Physics 4
Part VI: The Part of Tens 4
Icons Used in This Book 4
Where to Go from Here 5
A Final Word 6

Part 1: Basic Flash 7

Chapter 1: Why You Want to Write Games in Flash 9

Designing and Writing Games 10
Making artificial worlds 10
The importance of interactivity 11
Games are about objects 11
Players compete with the programmer 11
Game Programming in Flash 11
Comparing ActionScript with Animation 12
How You Make a Game 13
Making a playable game 13
Starting with a plan 14
Learning to code 15
Game Programming 101 16
Selecting a language 16
Planning tasks 17

Chapter 2: Cruising and Using the Flash Environment 19

Creating a New Program Project 19
Examining the layout of the Flash environment 21
Writing on the Flash Stage 24
Testing your program 26
Making a Web page with your creation 27

Adding Buttons 29
Building a button 31
Introducing the Library 32
Adding state to your button 36
Finishing your button 36
Adding code to the button 37
Understanding the code 38

Part II: The Next Steps 43

Chapter 3: Altered States 45

State of Nonconfusion 45
Adding Keyframes 47
Building the Green Grass game 48
Modifying the second frame 50
Making a Great Adventure 53
Planning your game 54
Setting the stage 55
Making the game your own 59

Chapter 4: Getting with the Program 61

Different Text for Different Jobs 61
Static text 63
Dynamic text 63
Input text 63
Building the Greeting Program 64
Adding text fields to the Stage 64
Associating variables with text boxes 66
Changing a text box through code 67
Reading information from an input text box 67
On a Roll: Making Random Numbers 69
Introducing the Math object 71
Random acts of randomness with Math.random() 71
Getting a 0–5 value 71
Making a six-sided die 72
Making Decisions with Conditions 73
Rolling the die 75
Checking your 6 76
Building the condition 76
Responding to False Conditions 78
Seeing the flaw in gotSix 78
Using the else clause 79
Making Lots of Decisions 81

Chapter 5: Making an Interactive Game 85

Introducing the Math Game 85
Making an Adder 87
Building the Visual Design 90
Designing the choose page 91
Designing the solve page 92
Designing the report page 94
Coding the Pages 95
Coding the choose page 96
Coding the solve page 99
Coding the report page 103
Coping with Bugs and Crashes 106
Syntax error 106
Nothing happens at all 107
Statement must appear within onClip event handler 108
The program moves to the score frame, but you don't see the plus sign 109
Something else is wrong 109

Part III: Sprites, or Movie Clips 111

Chapter 6: Introducing Sprites and Movie Clips 113

Building a Sprite 113
Making a movie clip 114
It's alive! Adding motion to your movie clip 117
Don't Object to Objects 120
Properties 120
Special functions 121
Characteristics 122
Making a Well-Behaved Object 123
Adding dx and dy properties 123
Building the onEnterFrame event 125
Moving the ball OOP-style 126
Overcoming Your Boundaries 128
Boundary effects 128
Combinations 136
Making a Cursor 137

Chapter 7: Won't Be Long 'Til You Write Pong 141

Building the Game Plan 142
Following the Mouse with the Player Paddle 144
Adding the Bouncing Ball 147

Building a Better Bounce 150
Risk has its rewards 151
Refining the bounce 152
Getting a new dy value 153
Adding a Computer Opponent 156
Building Artificial Stupidity 158
Adding a Scorekeeping Mechanism 161
Adding scorekeeping text fields 161
Add the scorekeeping code 163
Add starting, winning, and losing states 165
Making other states 167
Adding code to handle states 169

Part IV: Getting Control of the Situation 173

Chapter 8: Keyboard Input and Audio Output 175

Introducing the Monster Traffic Game 175
Responding to the Keyboard 178
Trolling for key presses 179
Examining keyboard input 179
Working with the Key object 180
Adding a keyboard handler 183
Adding Sounds 183
How Flash sound works 184
Getting sound effects 185
Considering audio compression 186
Importing a sound into Flash 187
Incorporating sound into your programs 191
Getting the most from your sounds 191

Chapter 9: It's Alive! Animating Your Sprites 193

Creating Animated Sprites 193
Building a shape 193
Building an animated sprite 198
Moving a Sprite under Computer Control 201
General plan for moving sprites 201
Setting up direction constants 203
Determining sprite properties 204
Turning a sprite 206
Moving the sprite 211
Animating the car 211
Creating a User-Controlled Sprite 212
Planning keyboard input 213
Checking for motion keys 214
Controlling the monster 216

Chapter 10: Building the Monster Traffic Game 219

Reviewing the Basic Design 219
Adding More Opponents 220
Cloning the movie clips 221
Coding for multiple enemies 221
Firing Missiles 223
Testing for Collisions 228
Planning your collisions 229
Adding collision code to your game 229
Building the checkCollisions() function 231
Building the ResetFlame() function 232
Adding the Sound Effects 234
Completing the Program 236
Adding an intro frame 236
Create the other states 239
Adding the scorekeeping functionality 240
Adding the animations 242

Part V: Phun with Phuzzy Physics 245

Chapter 11: Vectors and Gravity 247

Tower, Give Me a Vector 247
Working with vectors 248
Examining the vector 249
Making a triangle 250
Seeing things the trig way 250
Getting help from Chief SOHCATOA 252
How do I get dx and dy? 252
Going the other direction 255
Doing Vector Conversion in Flash 256
Introducing the vector projection demo 257
Calculating the values 258
Using Vector Projection in Motion 259
Building a cannon 260
Reading the keyboard 262
Moving the bullet 263
Turning the bullet 264
Fun with Ballistics 266
Understanding the gravity of the situation 268
Drawing on a movie clip 270
Drawing the path 271
Calculating the Vector from dx and dy 272
Determining the angle 274
Determining the vector length 275

Following the Mouse 275
Programming the EnterFrame event 276
Building the followMouse routine 276
Responding to the mouse click 278

Chapter 12: Vehicle Motion 279

Newton without the Figs 279
Newton's First Law 279
Newton's Second Law 280
Newton's Third Law 282
Newton and Vectors 283
Empty balloons fall to Earth 283
Adding helium to the balloon 284
Bringing wind into the mix 285
Don't tie me down 286
Baby, You Can Drive My Car 287
Checking keys for vector input 290
Turning the car 291
Making an object-oriented car 291
Making an even better car 295
Coding the parameter car 297
Getting Lost in Space 301
Building a multi-state sprite 302
Initializing the ship 303
Checking for input 303
Turning the ship 304
Moving the ship 306
Captain, We're Caught in a Gravity Well 306
Creating the universe 307
I'm pulling for you 308
If one planet is good 310
Building a Better Boat 312
The Secret of Traction 313

Chapter 13: The Life and Death of Sprites 317

Here We Go Loop-de-Loop 317
Making Many Things with Arrays 320
Building Sprites Dynamically 323
Dynamically generating a sprite 324
Building a suicidal sprite 326
Making many copies of a sprite 328
Creating Custom Objects 331
Making a really simple object 332
Building custom sprite objects 334
Using a custom movie clip class 335
Building a custom movie clip 336
One loop to control them all: Making many custom movie clips 340

Part VI: The Part of Tens 341

Chapter 14: Ten Math Concepts for Game Programmers 343

Managing Velocity 343
Accelerating an Object 344
Calculating a Distance 344
Projecting a Vector 345
Generating a Vector 346
Compensating for Gravity 347
Newton's Second Law 347
Generating a Random Integer 348
Combining Vectors 348
Sophisticated Vehicle Motion 349

Chapter 15: Ten Game Starters 351

Asteroids 352
Building Asteroids 352
Enhancements to Asteroids 353
Lunar Lander 353
Building Lunar Lander 354
Enhancements to Lunar Lander 355
Egg Cannon 355
Building Egg Cannon 356
Enhancements to Egg Cannon 357
Zelda 358
Building Zelda 359
Enhancements to Zelda 361
Platform Scroller Games 362
Building a platform scroller game 363
Enhancements to a platform scroller game 364
Breakout 365
Building Breakout 365
Enhancements to Breakout 366
Space Invaders 367
Building Space Invaders 367
Enhancements to Space Invaders 368
Orbit Matcher 369
Building Orbit Matcher 369
Enhancements to Orbit Matcher 370
Tile-Based World Games 371
Building a tile-based world game 372
Enhancements to a tile-based world game 373
Whack-an-Author 373
Building Whack-an-Author 374
Enhancements to Whack-an-Author 375

Index.....377

Introduction

I'm sure you bought your computer to do all kinds of serious work. Computers are good for homework, e-mail, work, and other perfectly respectable endeavors. But face it: Computers are also all about games. I love games, and I always have. As soon as I started to learn about computers, I wanted to use them to play games. I soon found it even more fun to make games than to play them. Even though I have a (somewhat) respectable career as a computer science teacher, the gaming aspect of computing has stayed with me.

If you're like me — with a love of games and curious how to write them — this book is for you. Most books on computer programming are pretty boring, but not this one. For example, I show you how to blow up stuff (as in *Kaboom!,* not as in enlarging a photograph). Most books on computer gaming are really technical, with endless descriptions of graphics primitives and indecipherable function calls. Not this one, though. I get things going as quickly as possible and let Flash do all the dirty work.

Yup, you read right, *Flash.* The Flash environment has emerged as a terrific tool for writing Web-based games. I dedicate this book to how games are made using this terrific tool. Along the way, you can glean some skills that might be useful in more ordinary programming contexts, too.

Okay, geek-speak disclaimer: Sometimes I have to use geeky words and even a little (gasp) math. Don't worry, though. Everything I show you has a purpose, and there won't be a quiz later. I speak English, too, so I promise to explain everything in regular English, with lots of fun analogies. (My favorite is the dog that does trigonometry.)

What's Really (Not) Required

If you're not sure you know everything you need to get started, don't worry! Here's what I *don't* assume you know upfront:

- **I don't expect you to be an ace computer user.** You should, though, be comfortable with all the ordinary computer operations, like saving/loading files and getting around in your operating system.
- **You don't need a super-high-speed computer.** Any system that can run Flash MX 2004 will do. These games work on even more humble machines.

- **You don't have to be a Flash master, either.** If you know how to make really great Flash animations, that's wonderful but not really necessary. Game programming is different from animation.
- **You don't need the most expensive version of Flash** (Flash MX 2004 Professional). This version of Flash does add some special features, but you really don't need any of those features to write wonderful games.

  This book was written using Flash MX 2004 with the latest updates available. If you're running an earlier version of Flash, some of the programs will still run, but you won't be able to open the FLA files from the Web site.

- **You definitely don't need to be a pasty-faced, mega-caffeine-swilling computer programmer.** (However, if that describes you, you're still going to have a great time, you l33t haxor!) I start from the very beginning, using game programming to teach the basic tenets of programming in any language. ***Teaser:*** Stick around for more catapulting cows here than in any COBOL book you've ever seen.

So what is required? Only a copy of Flash MX 2004, some determination, and a lot of imagination.

About This Book

Each chapter in the book describes a particular facet of game development. You can read the chapters in any order you wish, especially if you already have some knowledge of Flash or programming. If you're just starting, however, I recommend reading this book from front to back, simply because programming is a cumulative skill.

If you want, you can just download files from the Web site and start playing away. Most of the examples in the book are much more interesting in real life than I can show in a screen shot. Keep in mind that most of the example games on the Web site are left very simple to illustrate one particular idea. Still, they are pretty fun, and after you play them, I bet you'll want to read how they were made so you can change them and make your own variant.

Another fun alternative is to start at the very last chapter, which shows how to write ten different styles of games. Choose a game type that you want to master and go back to those chapters you'll need to pick up the necessary skills. This approach allows you to get to the game you want quickly without having to wade through anything that doesn't relate directly to that game.

How This Book Is Organized

I organized this book by writing a sophisticated Bayesian filter, artificial intelligence algorithm. Just kidding. Really, I sketched it on a napkin at the breakfast table. Still, I think it makes sense to break the book into a series of sections.

I lovingly named these parts as follows.

Part I: Basic Flash

This part gives you a programmer's introduction to the Flash environment. You see the various doohickeys and thingamabobs on the screen — and which ones you can ignore. You read how to make text appear and change onscreen, how to respond to button presses, and how to build a basic adventure game.

Part II: Getting with the Program

Time to experience some traditional programming skills (but nothing too boring). In this part, you master text-based input and output, see how to build random numbers, and make the computer perform the basic mathematical operations you'll use to build space muskrats in later games in the book. I show you how to make a sophisticated math game that generates random math problems. After that, I promise — no more educational games.

Part III: Sprites, or Movie Clips

Here you can use the most important element in Flash: the movie clip. Read here to find out what a sprite is and how you can use movie clips to make them easily in Flash. Then see how to build and control basic movie clips, making them move around onscreen, bashing into walls and each other. For a little ramble down Nostalgia Road, stick with me here to build the all-time classic *Pong* game.

Part IV: Getting Control of the Situation

Games aren't much fun if the user doesn't do anything. This section shows you how to respond to keyboard input and control sprites onscreen via player input. You also see how to add sound effects to your games (so anybody playing your game at work runs the risk of being fired). You discover more

sophisticated ways of moving and animating your sprites to make them more realistic. Follow along as I walk you through building a complete game — Monster Traffic — complete with monsters, flames, car alarms, destruction, and mayhem.

Part V: Phun with Phuzzy Physics

Don't worry — this isn't anything like Physics 101 with Professor Baldnoggin. Oh, no. The stuff in this section is much more cool than that. Sure, I've got to use words like *mass* and *vector* at some point, but it's worth it because you use these ideas to build vehicles that turn realistically, boats that skid around on water, spacecraft that orbit planets realistically, and all kinds of other geeky fun. You also become the true master of your universe as you see how to create and destroy sprites at your slightest whim (Muhahahaha!).

Part VI: The Part of Tens

The famous Part of Tens is a staple of any book in the *For Dummies* series. The two chapters in this part are pretty handy. The first one outlines the ten most important math concepts for a game programmer. These are ideas that you see throughout the book. Master these, and you master game development in any language. The last chapter is my favorite in the whole book. I wrote starter code for ten different games. I didn't finish any of them — that's your job! I did get the basic framework down so you can add your own flourishes. You'll find several classics (such as *Space Invaders, Zelda,* and *Asteroids*) and a couple of original ideas. You can think of this section as a recipe book to get you started on your own games.

Icons Used in This Book

Certain concepts in any book ought to stand out on the page. With that in mind, this *For Dummies* book includes a number of margin icons for certain situations:

Tips are suggestions to make things easier.

Sometimes I have to talk about certain technical things in order to keep my *Self-Important Computer Science Instructor Certification.* These things are interesting but not crucial, so I mark them with this icon. You don't need to read them if you don't want, but memorize some of these paragraphs before you go to your next computer science party. The guests will love you.

Be sure to read text marked with this icon! If you do not follow a warning, bad things could happen: Puffs of black smoke might come out of your monitor, your workspace could be deluged by a plague of frogs, or your program simply won't work right.

These tidbits denote info you ought to think about, but it's not going to cause a disaster if you don't pay attention.

If you're gonna be a programmer, you gotta have code. Of course, I give you all the source code files for this book, located handily online at

```
www.dummies.com/go/flashgameprogrammingfd1e
```

Where to Go from Here

My recommendations on how to proceed? Mainly, have some fun and write some games.

- Begin by simply downloading the software and playing the games I've put there.
- If you're really new to all this stuff, jump in and start writing games. I put the easiest game programming tasks at the beginning, but you can start wherever you want. If you start in the middle and get confused, just back up until you're comfortable.
- For all other concerns, use the index or jump straight to the chapter you need. (You can always return later at your leisure.)

A Final Word

Thank you for buying this book, and I hope that you find *Beginning Flash Game Programming For Dummies* fun and valuable. I had a great time writing this book, and I think you'll have a lot of fun using it to write really terrific games. Have fun, learn a lot, and let me know what you've made!

Part I
Basic Flash

In this part . . .

You discover the basic toolset of the Flash environment as a programmer sees it. You give your programs various states and take a tour of all the beginning tools. You finish the section with a complete adventure game.

Chapter 1 shows you how to start thinking like a programmer. I explain how Flash and ActionScript are like other programming languages and some key ways they are different. If you've never programmed before, I prepare you with some wisdom about the programming process.

Chapter 2 is about creating Flash projects. You make a button and have it respond when the user clicks it. You find out how to embed your Flash games into Web pages.

Chapter 1

Why You Want to Write Games in Flash

In This Chapter

- Seeing how to use Flash to write games
- How programming differs from animation
- Exploring basic game design concepts

Computer programming can be a whole lot of fun. That's why I got into it way back when, and it's why I still do it. Truth be told, the main reason I learned how to program was to write games. I couldn't buy much software for my first computer (a TRS-80 Model 1, still in the garage . . . sigh). I wanted to play games, so I had to create them myself. Admittedly, I was pretty bad at it, and I failed a lot, but I kept trying. As I grew up, my programming skills were marketable in the "serious" world, but I never lost my fascination with computer games.

Here are some very good reasons to write games:

- Computer games made more income in 2003 than the movie industry.
- Game programming is technically challenging.
- Making a game is *fun!*

Most other game development books can be divided into two camps:

- Some talk about the game design process, storyboarding, coming up with game ideas, and the visual side of gaming. That's pretty good stuff to know, but it doesn't help you actually make a game.
- Other books assume that you're already good at C++ and advanced math. That's pretty good stuff, too, but you don't need to start there.

I believe that newcomers to programming can master the essential ideas of programming at the same time they're learning to build games. I also feel that those with some programming experience will truly enjoy the uniquely creative aspects of game development. You don't have to know anything about programming or Flash to use this book. (However, if you know these things, you'll still probably see something new.)

In this chapter, I give you an overview of the basics of game designing and planning, writing, and programming in Flash (with ActionScript). Most of all, you're going to have a lot of fun.

Designing and Writing Games

If you've asked around about how to get started in game programming, people have probably told you to learn C++ and take lots of math classes. That's not bad advice, but I have an easier way. The truth is that making games isn't really about any particular computer language. After you learn how to write games, you can transfer those concepts to any environment you wish. There are surprisingly few main concepts behind game development. If you truly understand these ideas, you can translate them to any programming language you want.

In this book, I show you how to program games in Flash. I like Flash because it simplifies the visual side of programming, works on almost every computer made, and has a powerful and reasonably easy programming language. I talk about this more in the upcoming sections, "Game Programming in Flash" and "Game Programming 101."

Too, game programming is different than other kinds of software development. For one thing, games need to be fun. And games are all about communicating with the player as well as providing some sort of immersive world in which the player participates. As a game programmer, you get to be creative and think outside the box.

Making artificial worlds

Typical business programming relies heavily on certain conventions and metaphors. If you're writing a database application, it's *de rigueur* to make your program much like all the other programs users have seen. In game programming, though, you're often trying to "hide" the computer from the player. For example, if you're making a spaceship game, you want the controls to look and feel like spaceship controls. Imagination is a really important part of playing and writing games.

The importance of interactivity

Games need to react to the player. The player should manipulate a virtual presence, and the game should react accordingly. Some games are turn based, and some are in real time, but all require more immediate feedback than traditional types of programs.

Games are about objects

Many games involve objects bonking into each other, shooting each other (with other objects), avoiding each other, and simply milling around. While you're writing a game, you usually think about objects, their characteristics, and how they interact with the player as well as with each other.

Players compete with the programmer

When you play a really great game, you're not really playing against the computer. Rather, you're really engaging in a stylized conversation with the programmers. As a game developer, you get the chance to set up worlds. The players interact with a stored version of your thoughts and imagination.

Game Programming in Flash

Macromedia Flash is a very good environment for learning basic game programming ideas. Here are a number of reasons for starting with Flash:

- **Flash offers robust multimedia support.** Flash, which was designed to support animation on the Web, supports various kinds of images easily. (Think JPEG images and custom drawings.) See how to use the drawing features of Flash in Chapter 9. Flash also has great support for various kinds of audio files, such as MP3 and WAV formats. You incorporate audio into your games in Chapter 8.
- **ActionScript is related to the influential C language.** The ActionScript programming language built into Flash is closely related to JavaScript and ECMAScript, which are two extremely common programming languages. All these languages are based on the C programming language, so the coding conventions you'll master are much like those in other languages.

- **Flash is designed for the Web.** By working in Flash, you have a ready distribution network. Because Flash was designed for the Web, all your games can be easily published on the Web, and anybody with a Web browser and a Flash plug-in can enjoy your games. And you won't have to worry about what operating system your users use. (All the programs in this book have been tested in Windows XP and Fedora Core Linux, but they should work in any OS with a Flash plug-in.)

Comparing ActionScript with Animation

Maybe you've used Flash to build Web animations without ever going into its programming features. Many books on Flash (as opposed to ActionScript) focus on the powerful animation features of Flash. These books often mention ActionScript but don't dwell on it heavily. Animation is primarily about creating moving images; user interaction in animations is minimal. When creating an animation, you generally create some sort of visual symbol onscreen and then use a tool called a *motion tween* to indicate where this object should be at a specified point in time. You can also use a tool called a *shape tween* to change the shape of an object over time. You can do this with many objects at the same time to make a complex animation. In order to track all these objects, Flash animators often arrange them into separate *layers.* Thus, a typical 30-second Flash animation might have hundreds of frames of animation in over a dozen layers.

Animation is cool because it allows you to build movies. However, to create games, you must discover how to program.

If you treat Flash as a programming environment (as I do in this book), you see things quite differently. You still use Flash to create objects, but instead of relying on the Flash environment to control what those objects do (via animation), you control the objects directly by writing programming code. The ActionScript programming language built into Flash lets you do anything that can be done with animation — and many things that cannot be done by using animation techniques alone.

In a nutshell, programming is what makes games interactive. You can

- Control what's onscreen, what size it is, where it is, and how it's rotated
- Detect whether two things touch each other
- Accept input from the user

How You Make a Game

The goal of game development can be summarized in one sentence:

Games are stories that use the player as a primary character.

Like any interesting story, a game needs these plot elements:

- A character (at least one)
- A conflict
- A goal

Game play must be compelling, but game elements don't need to be complex. Simple games like *Tetris* and *Pac-Man* have had phenomenal success.

Making a playable game

A good game has a good story, and it also has some form of user interaction. In Flash, the player uses the mouse and keyboard as primary input devices. Although these devices might seem limiting (compared with a more sophisticated joystick or driving console), you can do many things with these basic forms of input.

Although Flash doesn't directly support joystick input, users can easily use modern joysticks with the games you can write in Flash. Most joysticks now come with programs that allow the user to map keystrokes to keyboard commands. In effect, by allowing keyboard input, you also allow rudimentary joystick input.

A game should also look good and sound good, but these things don't matter if the game isn't fun.

Some of the best games ever have incredibly limited graphics and sound. If you've never played *NetHack* (as shown in Figure 1-1), download a copy (free for just about any computer ever made) and play it. At first, you might be thrown by the complete lack of graphics and sounds. *NetHack* uses plain text without any graphics or sound effects, but the game is amazingly absorbing. I bet that you get caught up in the incredible game play and find yourself actually scared of the capital D coming at you.

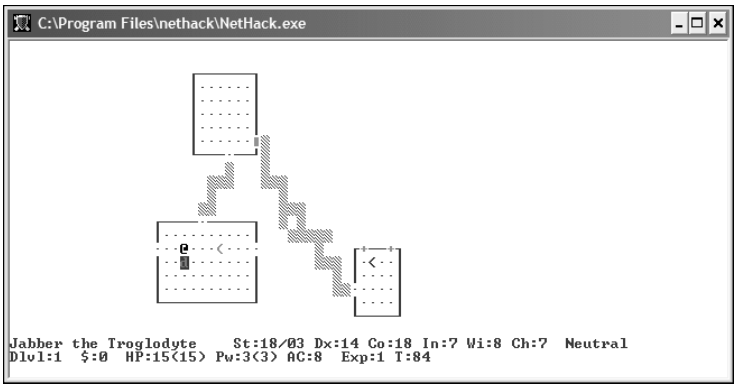
C:\Program Files\nethack\NetHack.exe
Jabber the Troglodyte St:18/03 Dx:14 Co:18 In:7 Wi:8 Ch:7 Neutral
Dlvl:1 $:0 HP:15(15) Pw:3(3) AC:8 Exp:1 T:84

Figure 1-1: A captivating game isn't always about flashy graphics and sound.

Most of all, games should be fun. I can't really tell you how to make a game fun. You need to test a lot for a game that's fun to play.

Starting with a plan

Before you worry at all about the details of your game, come up with a theme.

Think about what you want your game to be about. Outline and define the following components:

- **The main character:** Don't forget the kinds of obstacles this character will encounter. You can read how to build a main character throughout the book, but the topic is covered most deeply in Chapter 9.
- **The overall look of the game:** Consider the setting. What colors will you use? What overall look and feel are you looking for? (Retro? Cartoon? Gothic? Maybe a Gothic retro cartoon?) Chapter 7 describes how to set up the visual feel of a game.
- **The main screens:** Most games have
  - A main play screen (or two)
  - An instruction screen
  - Some sort of introduction
  - A Game Over screen, or maybe two: one for when the player wins and one for when the player loses

  Chapter 7 describes how to build multiple screens for your games.

  Draw these visual elements on paper.
- **The objects on each screen:** You have to build everything in your game. The visual design part is important but relatively easy.

- **The role/behavior of each object:** Decide these details upfront for each object:
  - How the object moves
  - Whether it's controlled by the user or the computer
  - Whether it does something when it interacts with other objects
  - Whether it makes sounds
  - What happens when it leaves the screen

After you finish defining these objects, convert your sketches into reality. This sounds like a pretty easy step, but it's the one that might cause you a lot of grief. You probably know exactly what you want all the screen objects to do, but a computer is incredibly stupid.

You have to convert your clever ideas to statements so clear that even an idiot computer can understand them.

Learning to code

Mountain climbers train before they scale the big mountains:

- **Learn:** They learn the tools of the trade, practicing on small hills and isolated, safe areas before testing their skills on actual mountains.
- **Pace:** When they're ready to climb Mt. Everest, climbers don't go for the top in one day. They build a solid base camp at the foot of the mountain. Then they create another camp higher up, and another even higher.
- **Progress:** At each camp, the ultimate goal is still the summit, but the intermediate goal of the next camp is the task at hand.

  A mountain climber concentrates on the next step.

The same advice is really good for all programmers, beginning or advanced:

- **Master the tools of the trade.** There's no getting past the fundamentals (which this book shows).

  You need to know both
  - The basic ideas of programming
  - The principles of game development
- **Know the goal.** That's why you start with a written description of your program.

- **Use small steps.** Concentrate on mastering one task at a time.

  In this book, chapters show you specific skills and apply those skills in simple games, so you can

  - Learn the skills you need for complex games.
  - Practice these skills in isolated programs.

- **Enjoy the view.** Game programming is supposed to be fun.

  Celebrate your progress! When you succeed at a viable chunk of code, do a little Hampsterdance. (If you don't know what I'm talking about, visit `www.hampsterdance.com`.)

- **Pace yourself.** Your first program won't be the next version of *Quake,* but there's plenty of fun in writing games that are a little less ambitious. Eventually, you'll build your skills so you can write something way better than *Quake.*

Game Programming 101

Game programming is a process. All the programs in this book use the Flash environment, but the details of Flash programming aren't the most important factor. When you want to make a game, you need to choose an environment that will work as well as decide a strategy for creating the game.

Selecting a language

If you're reading this book, you've chosen Flash as your environment. Excellent choice!

- **Flash is an ideal environment for beginning game creation.**

  Flash makes a lot of the implementation easier, so you can concentrate on the content of your games instead of all the details of memory management, image drawing, and reading the input devices. (Fancier environments make you put a lot of work into such details instead of mastering the craft of game development.)

- **Most commercial games are written in 2-D, using C++ and graphics engines like DirectX or OpenGL.**

  Those are really great environments, but they aren't necessarily what you need while you're learning the process of game development.

If you want to be a racing champion, you don't just show up in Indianapolis with a helmet. Starting your driving career in a high-performance machine is foolhardy and dangerous. You begin your career racing karts and then advance through more challenging vehicles. That's why you should start programming with Flash and ActionScript:

- C++ is like a Formula 1 car — fast and difficult to handle.
- Flash and ActionScript are like a go-kart (albeit a souped-up, Internet-enabled go-kart that outperforms any computing environment NASA had during the moon program).

Planning tasks

Game programmers prepare by planning several parts of the game:

- **Encapsulating objects onscreen:** All the things that move around on the computer screen are called *sprites* by game programmers.

  Chapter 6 shows you what need to know:

  - What a sprite is
  - How to create a sprite with Flash tools

  Flash has a great object called a *movie clip* that can easily be used as a basis for sprites. Chapter 6 shows how to use it.

- **Accepting input from the user:** There are all kinds of user input devices, but Flash games concentrate on the mouse and the keyboard.

  Chapters 2 and 6 show how to get information from the mouse, and Chapter 8 describes reading the keyboard in Flash.

- **Moving things realistically:** Game programmers must

  - Understand the physics properties of position, velocity, and acceleration, and attach these characteristics to a sprite so that it moves in a realistic fashion.
  - Simulate such useful physical properties as gravity and friction.

  Chapter 6 shows how to manage basic motion in Flash. Read about more sophisticated motion in Chapter 9 and see how to create realistic vehicles in Chapter 12.

- **Dressing up the user's experience:** Graphics, sound, and animation matter. Chapter 9 shows you how to use graphics, and Chapter 8 shows you to use sound effects.

- **Keeping the action fun:** Every game must be able to adapt to the user's ability level.

  Find places to adapt the computer's ability so the computer always gives users an interesting challenge. As I describe each game in the book, I give hints how you can make the game easier or more difficult.

Chapter 2

Cruising and Using the Flash Environment

In This Chapter

- Picking a project to start with
- Checking out the work area
- Following the `Hello World!` tradition
- Adding buttons to your programs
- Creating various button states
- Responding to button events
- Publishing your work

The Flash development environment has a lot of powerful features that can be intimidating to a beginner. For example, the default screen has 12 different panels with over a hundred buttons, icons, and menu choices. Fortunately, there's a lot of stuff you won't have to worry about, and the following sections present the essential interface features — such as the Stage and Timeline — so that they begin to make sense very quickly. The best way to find out about the Flash MX environment is to use it to write some programs. This chapter helps you build a basic adventure game, and along the way, you get chummy with the basics of the Flash interface.

Creating a New Program Project

The ideas in this chapter can be put together to build the classic first program, one that declares your programming prowess to the world. Figure 2-1 illustrates my take on this masterwork.

Figure 2-1: This is a very friendly program.

When you first open Flash, you see a screen much like the one shown in Figure 2-2, which displays a list of project templates that you can create. The choice that you make here determines the starting characteristics of your project, but you can change most of the settings after you learn to manipulate objects and their properties.

To begin a new game project, just click Flash Document in the Create New column.

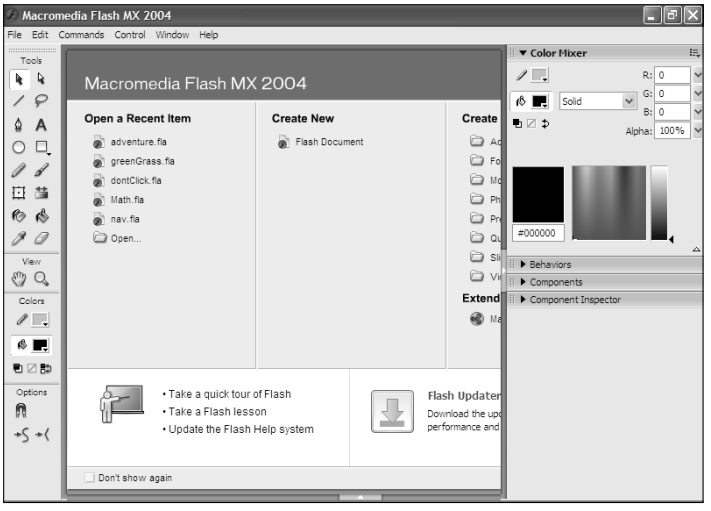

Figure 2-2: Build a new Flash document to get things started.

The basic Flash Document project provides the foundation for everything you do to create your game program. From this base document, you can

- Add buttons, text boxes, and game objects to populate the game interface.
- Draw and color basic shapes that are the foundation of your games.
- Move your game's objects around in the game staging area.
- Control how your objects react over time.
- Respond to user input from the mouse and the keyboard.
- Play sounds.
- Create anything else you typically see in Web-based games.

The other templates simplify the creation of other kinds of programs, including projects optimized for mobile devices and video applications. You can build a game designed to fit on a Pocket PC, a Palm Pilot, or the new generation of cellphones! There's nothing really that fancy about the templates. Everything you can build with them can also be created with a normal Flash document. The templates are simply preset for specific kinds of projects. For example, the Mobile Device templates are designed to be the size of typical PDA and mobile phone screens.

Examining the layout of the Flash environment

As you begin your project, you see a screen like the one shown in Figure 2-3. A Properties Inspector, (bare) Stage, Tools panel, Timeline, and panel stack are at your disposal.

The following sections show how you use each section of the work area.

Stage

Most of the Flash work area is taken by a large rectangle near the center of the screen. This area is the *Stage,* where the action of your game takes place. After you create your games, the user sees only the Stage and the objects that you create bouncing around and interacting on it.

The term *Stage* is from Macromedia's earlier product *Director,* which took a moviemaking analogy to somewhat ridiculous extremes.

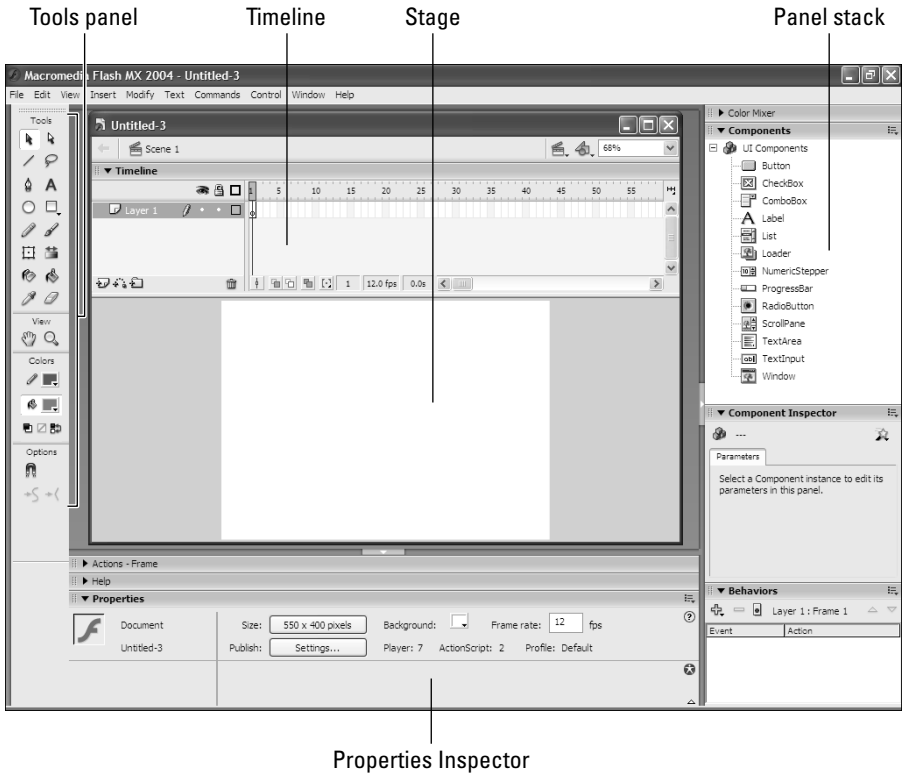

Figure 2-3: The Flash MX screen is divided into several major sections.

Properties Inspector

The Properties Inspector box is directly below the Stage. The *Properties Inspector* allows you to change characteristics of the things (such as the buttons, enemy objects, and the player's character) that populate your game. The Properties Inspector also changes automatically to reflect whatever object you're working with.

When the Stage is selected, the Properties Inspector box shows things that you can change about the Stage, such as the size and background color. When you create other objects (space monkeys or whatever) for your games, you use the Properties Inspector to change certain characteristics of these new objects.

Tools panel

The *Tools panel,* which goes along the left side of the screen, features tools for drawing and manipulating basic graphics. You use these tools to create the visual representations of the characters in your game.

Hello, World!

Looking around the interface is nice, but you're here to do some programming! May I suggest that your first program reflect one of the oldest traditions in programming? (Programming has all kinds of tradition and folklore, despite its relatively short time on the scene.) When programmers encounter a new programming environment, they often try it by writing a program that simply says, "Hello, World!" It's a fun tradition, but it also has a practical side. The `Hello World!` program has these benefits:

- It's about as simple of a program as you can write.
- It provides some kind of feedback (or visible output).
- It demonstrates a basic understanding of the mechanics of your programming environment.

There's no point in writing a more complex program until you get down the very essentials. Make sure you can write a minimal program that runs correctly and proves its existence by displaying some output. All other programs will be based on this foundation, so make sure you understand it before you add more meat to the program.

The tools are similar to common painting and drawing programs.

Timeline

The *Timeline* is a grid-like feature at the top of the Flash editor screen. Here, you can work with keyframes and layers to help create and refine your project.

Keyframes

In Flash animation, the Timeline indicates how various elements react over time. As a game programmer, however, you use the Timeline a bit differently, breaking your program into little slices of time called *keyframes.* You repeat each keyframe indefinitely. The timeline for an animation is often thousands of frames long, but most programs take only a few frames because each frame repeats. Instead of relying on the Timeline to control how elements move around on the screen, your code does all the work.

Layers

When you view a typical Flash document, many timelines are stacked on top of each other. Each row of the Timeline represents a different *layer,* which is used to separate parts of the game.

As a programmer, you don't need layers as often as animators need layers. Your code performs most of the features that animators usually generate with layers.